Session 402

QuickTime for Professional Video

Tim Cherna
QuickTime Professional Video Group

Why Professional Video?

- QuickTime is a foundation for professional video applications
- Unique demands
  - High quality
  - High performance
  - Tight hardware integration
  - Consistent results

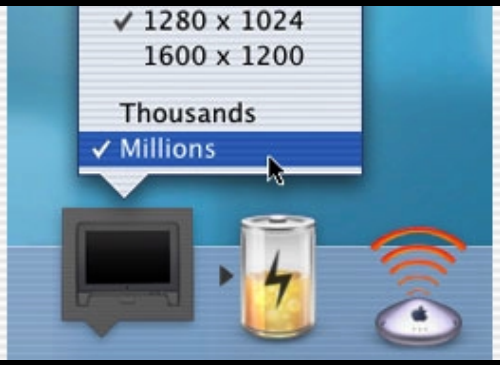
Who Are You?

- Application developers
  - Video editing
  - Video processing
- Codec writers
  - Hardware
  - Software

What You'll Learn

- Rendering improvements
- Hardware support improvements
- Multiprocessing and asynchronous operations
- Future developments in effects
- Video hardware and Mac OS X

Rendering Improvements

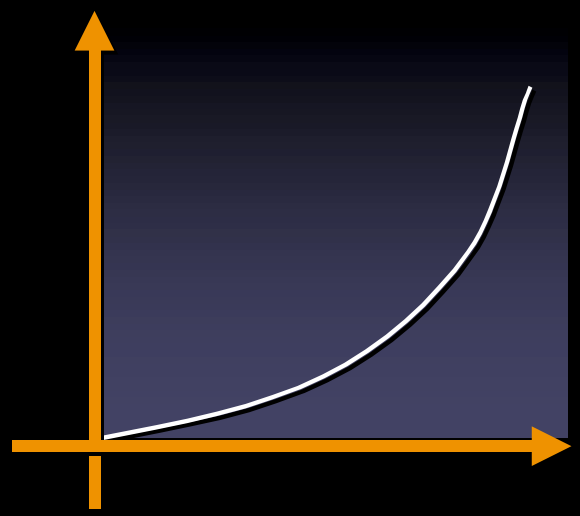


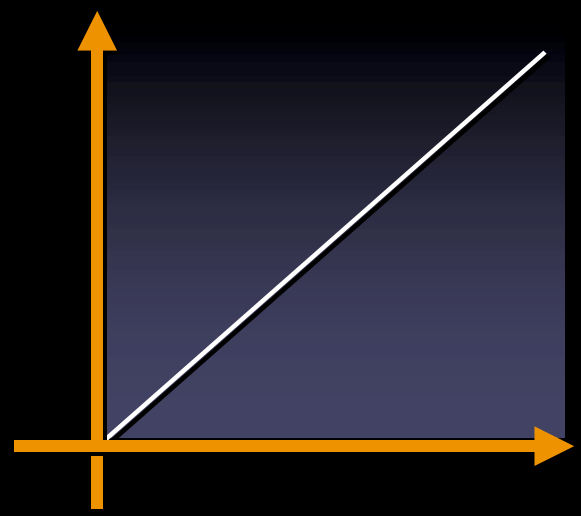
Rendering Improvements

- Better Gamma processing
- New r408 pixel format
- DV Codec improvements

Rendering and Gamma

- What is Gamma
  - Describes non-linearity of intensity reproduction
  - Input value/voltage => output intensity





Rendering and Gamma

- Problem
  - Native video space is 2.2
  - Macintosh graphics assume 1.8
  - Windows graphics assume 2.5
- Solution
  - Applications can specify gamma
  - Codecs can specify gamma
  - No more guessing

Rendering and Pixel Formats

- RGB
  - Native for graphics
  - Optional alpha channel
- YUV 4:2:2
  - Native for video
  - 2 Luma for every Chroma (subsampled)
  - No alpha channel

Problems

- RGB
  - Color space conversion required
- YUV 4:2:2
  - No alpha channel
  - Processing is hard for the Application
    - Black is Y = 16
    - Subsampled

r408 to the Rescue!

- Video friendly
  - YUV based
- Application friendly
  - Not subsampled
  - Has Alpha channel
  - Black is Y= 0

Better DV codec

- Gamma and r408 support
- Better quality
  - Consistent results on G3 and G4
  - Color fidelity
  - Much less generational loss
- Better performance
- Accelerated on MP Macintoshes

Demo

DV and Rendering

Hardware Support Improvements

Jean-Michel Berthoud
QuickTime Pro Video Group

Before QuickTime 5.0

- Assumption
  - All codecs can decompress “now”
- Problem
  - Hardware pipelines
- Solution
  - Make QuickTime aware of latency

Video Latency

- Codecs can report their latency
  - ImageCodecGetDecompressLatency
- Video starts “early”
  - Offset by latency
  - Movie starts when pipeline is full

Audio Latency, Too

- Same model for sound output components
- Sound output component can report latency
  - siOutputLatency selector in SoundGetComponentGetInfo

Before QuickTime 5.0

- Assumption
  - QuickTime chooses fastest codec
  - Codecs don’t lie about their speed
- Problem
  - They lie
- Solution
  - Allow application to choose codecs

Specifying Preferred Codecs

- Application tells QuickTime
  - MediaSetPreferredCodec
- Solution for application presets
  - Easier hardware configuration

Just One More Thing…

- Movies might require 3rd-party hardware codecs
  - Can be played on custom hardware
  - Vendors can supply software codecs to view content
- Problem
  - Users need to search for software codecs
- Solution
  - Let QuickTime automatically find them
  - Use Component Download mechanism

QuickTime on MP Macs

Sam Bushell
QuickTime Engineering

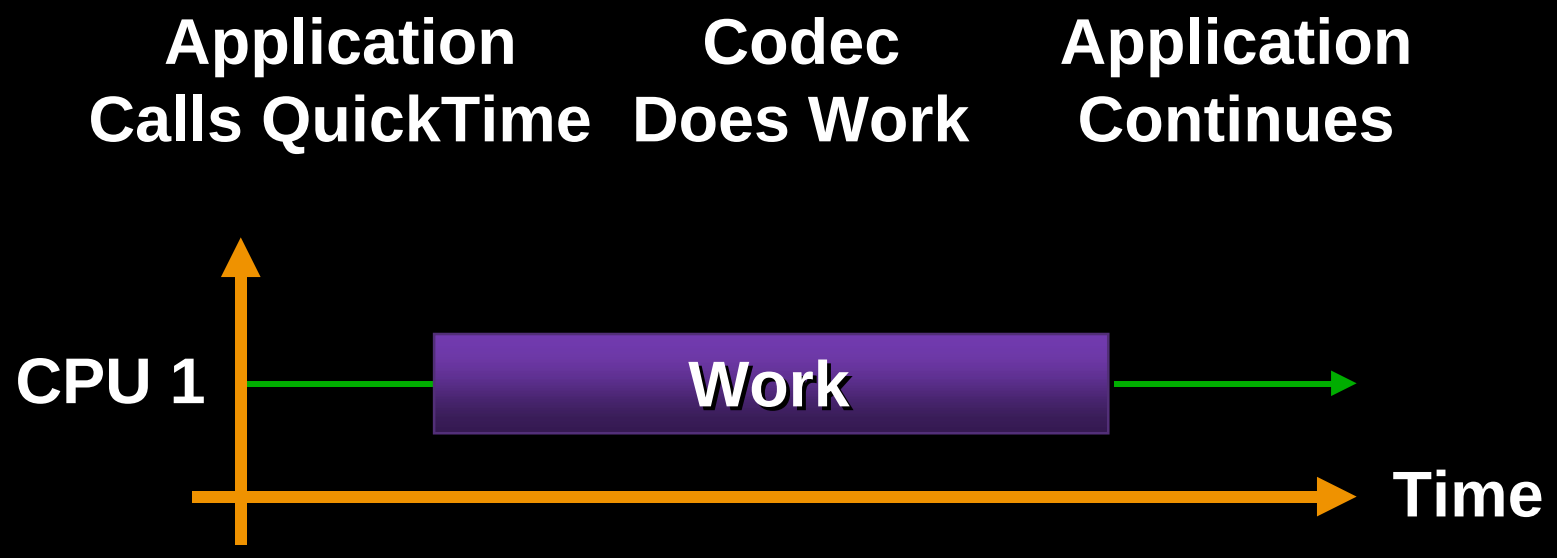
Two CPUs Are Better Than One!

- Right?

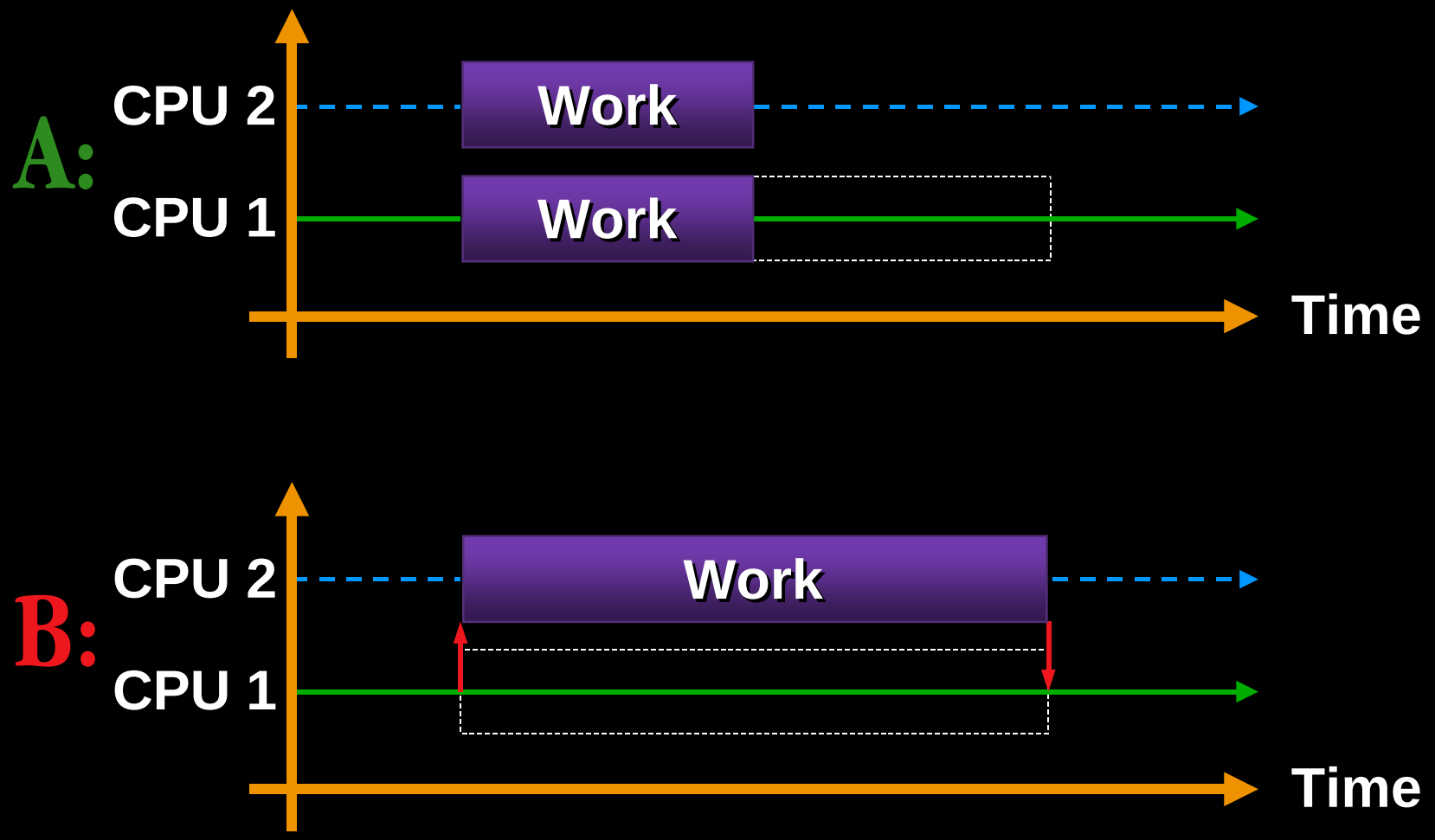
Two CPUs Are Better Than One!

- Only if they make things faster
- Sometimes the user runs several apps at once
  - Automatically faster on Mac OS X
- Sometimes only one app is doing work
  - QuickTime apps spend a lot of time in codecs
    - Compressors, decompressors, effects…

Single Processor

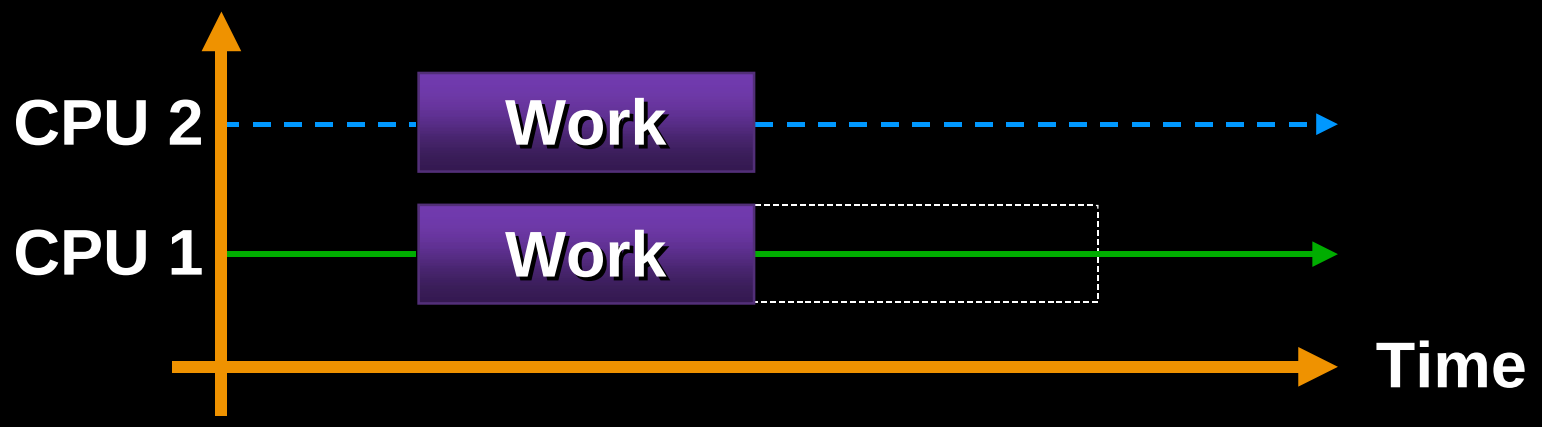


Two Approaches to MP



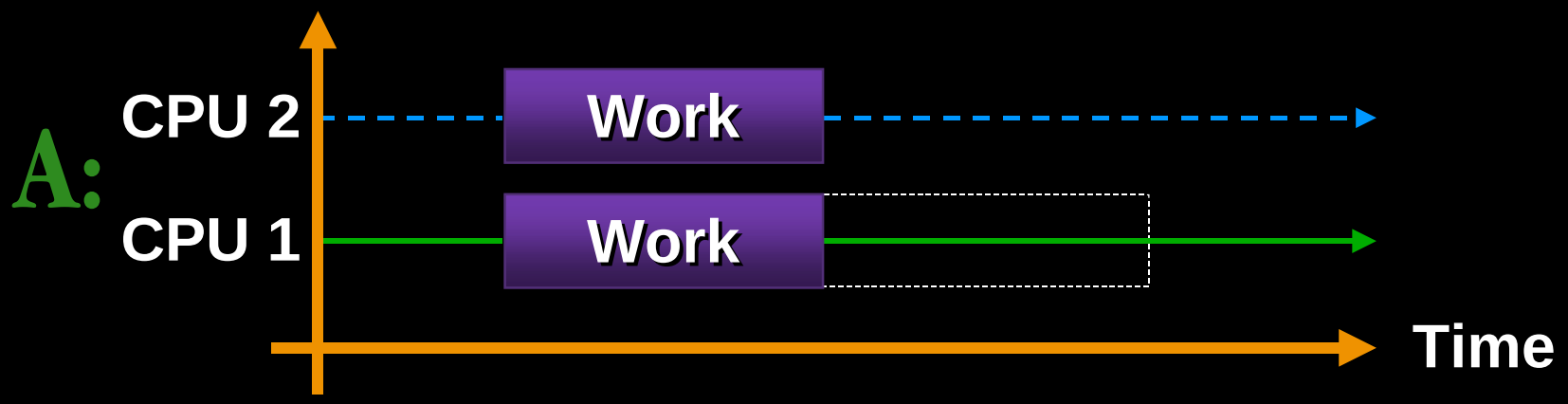
A: Split Up Across n Tasks

- Better
  - Works with existing applications
    - Synchronous APIs OK
  - Higher performance gains available



A: Split Up Across n Tasks

- Harder
  - Not feasible for all algorithms



B: Run Whole Job in a Single MP Task

- Easier
  - Smaller change to your code
  - QuickTime can help in some cases

B: Run Whole Job in a Single MP Task

- It takes just as long
  - But useful if application has something else to do
- ⇒ Must restructure applications to take advantage
  - Use asynchronous APIs

QuickTime 5 Uses Both Approaches

- Split up across n MP tasks
  - DV compressor and decompressor
- Run in single MP task
  - H.263 compressor
  - H.263, JPEG, Cinepak, Video decompressors

Demo

Compression at Multiple Bitrates

What Can You Do?

- Accelerate your application
  - Use asynchronous compression APIs
- Accelerate your codec
  - Approach **A** or **B**, it's up to you

Asynchronous Compression

- Existing Image Compression Manager API:
  - CompressSequenceFrame(
    ..., &completionProcRecord);

Asynchronous Compression

- Existing Image Compression Manager API:
  - CompressSequenceFrame(..., &completionProcRecord);
- New Standard Compression API:
  - SCCompressSequenceFrameAsync(..., &completionProcRecord);
  - SCAsyncIdle(...);

Accelerate Your Own Codec

- **A:** Split up across n tasks
  - Great, but you're on your own
- **B:** Run in single MP task
  - Base decompressor can help
    - Tell it that DrawBand is MP-safe

Pitfalls to Avoid

- Don’t allocate memory in MP DrawBand on Mac OS 9.x
  - Even on Mac OS 9.1 and later
  - MP RPC only serviced in WaitNextEvent

Pitfalls to Avoid

- Avoid approach **A** on Mac OS 9.0.4 and earlier
  - MP task page faults only get serviced in WaitNextEvent
  - Fixed in Mac OS 9.1

Pitfalls to Avoid

- Be careful with MP in decompressors that draw direct to screen
  - Tearing artifacts

Upcoming Effect Changes

Tom Dowdy
QuickTime Pro Video Group

Upcoming Effects Changes

- For effects developers
- For developers who use effects

New Optional Specifications

- Effect Major and Minor Class
- Effect Presets

Demo

New Effects UI

Effect Major and Minor Class

- Major Class used for filtering
- Minor Class used for UI grouping

Effect Major Class

- Used for Filtering
- Atom in effect description
- kEffectMajorClassType 'clsa', kEffectMajorClassID 1
- Must be one of:
  - kGeneratorMajorClass 'genr'
  - kFilterMajorClass 'filt'
  - kTransitionMajorClass 'tran'
  - kCompositorMajorClass 'comp'
  - kMiscMajorClass 'misc'

Effect Minor Class

- Use for UI grouping
- Atom in effect description
- kEffectMinorClassType 'clsi', kEffectMinorClassID 1
- Should be one of:
  - kGeneratorMinorClass 'genr'
  - kRenderMinorClass 'rend'
  - kFilterMinorClass 'filt'
  - kArtisticMinorClass 'arts'
  - kBlurMinorClass 'blur'
  - …continued

Effect Minor Class

- Should be one of:
  - . . .
  - kSharpenMinorClass 'shrp'
  - kDistortMinorClass dist'
  - kNoiseMinorClass 'nois'
  - kAdjustmentMinorClass 'adst'
  - kTransitionMinorClass 'tran'
  - kWipeMinorClass 'wipe'
  - k3DMinorClass 'pzre'
  - . . .continued

Effect Minor Class

- Should be one of:
  - . . .
  - kCompositorMinorClass 'comp'
  - kEffectsMinorClass 'fxfx'
  - kMiscMinorClass 'misc'
- Can also be custom via Pascal string
  - kEffectMinorClassNameType 'clsn'
  - kEffectMinorClassNameID 1

Effect Presets

- Atoms of type kEffectPresetType 'peff' and ID 1-N
- Contains three atoms to define the preset:
  - kPresetNameType 'pnam', kPresetNameID 1
    - Name of preset as a Pascal string
  - kPresetPreviewPictureType 'ppct', kPresetPreviewPictureID 1
    - Preview as a Picture, at least 86 x 64 pixels
  - kPresetSettingsType 'psst', kPresetSettingsID 1
    - Entire set of parameter values that define this preset

Migrating Hardware Components to Mac OS X

Jean-Michel Berthoud
QuickTime Pro Video Group

The Bad News…

- No direct access from component to hardware
- Need a driver layer abstraction
- Bringing up hardware and debugging is more complicated:
  - One piece lives in kernel space
  - The other lives in user space

The Good News…

- Existing driver layer or hardware access library will make it easier
- Creating one will maintain a Mac OS 9 version as well as moving to Mac OS X
- When you are done, you will probably never want to come back—memory protection is your best friend

The Mac OS X Driver Model

- Drivers belong to kernel space, components live in user space
- Use existing driver family (USB, FireWire, etc.) if you can
- If not, implement an IOKit library or CFPlugin
  - You become responsible for your own driver APIs

Big ‘No No’s on Mac OS X

- Read/write registers or physical PCI memory from your component
- Call QuickTime completion procs from your own interrupt handler
- 'hold' the all CPU within your components: threads will preempt you

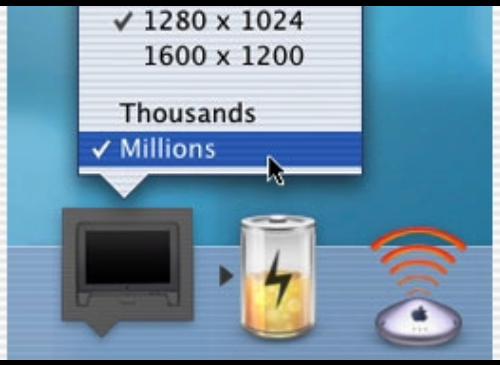
So How Do I Do That on Mac OS X?

- Access hardware only in your driver
- Use a thread in your component to call QuickTime completion procs
- Send kernel messages to component via Mach port events

Things to Know

- Crossing into kernel space is expensive
  - Minimize the number of calls
  - Especially for configuration APIs
- Threads are very cool but not free

QuickTime for Professional Video



Roadmap

403 Broadcasting Live with QT Streaming	Room A2 **Wed., 5:00pm**
406 QuickTime & Darwin Streaming Server Overview	Civic **Thurs., 9:00am**
407 Customizing QuickTime & Darwin Streaming Server	Civic **Thurs., 10:30am**
FF109 Feedback Forum: QuickTime	Room J1 **Thurs., 3:30pm**

Who to Contact

Jeff Lowe
QuickTime Technology Manager
Apple Worldwide Developer Relations
jefflowe@apple.com

For More Information

- **developer.apple.com/quicktime**
  - "What's New in QuickTime 5" doc
  - Ice Floes on rendering, pixel formats and effects
- C. Poynton, *A Technical Introduction to Digital Video*

ANNOUNCING

QuickTime Live!

October 8–11, 2001
The Beverly Hilton Hotel
Beverly Hills, CA

Full day workshops, Conference Sessions,
QuickTime Product Showcase, Keynotes and Social Events with the community
of QuickTime Developers, Content Creators, and New Media Business Professionals.
Get Face-to-Face with the QuickTime team for a week of training, business
Opportunities and industry perspectives that you can't get anywhere else.

www.apple.com/quicktimelive

Develop. Produce. Deliver.

WWDC

Worldwide Developers Conference 2001